REMARKS

This amendment is intended as a full and complete response to the non-final Office Action mailed October 6, 2004. In the Office Action, the Examiner notes that claims 1-20 are pending, of which claims 1-20 stand rejected. By this response claim 1 is amended and claims 2-20 continue unamended.

In view of both the amendments presented above and the following discussion, the Applicants submit that none of the claims now pending in the application are obvious under the provisions of 35 U.S.C. §103. Thus, the Applicants believe that all claims are now in allowable form.

REJECTIONS

35 U.S.C. §103 Claims 1-20

The Examiner has rejected Claims 1-20 under 35 U.S.C. §103(a) as being unpatentable over Mahler et al. (6,675,218, hereinafter "Mahler") in view of Garcia et al. (6,145,061, hereinafter "Garcia"). The Applicants respectfully transverse the rejection.

A. Claims 1-16

Independent claim 1 (independent claims 17 and 19 recite similar limitations) recites:

> "An application programming interface (API) for network applications capable of processing packets having source and destination node addresses different from a node where the application runs, said API comprising:
>
> first and second data structures associated with a network interface in communication with a network, said first and second data structures being mapped to an operating system and a network application, said network interface, operating system, and network application residing at a node capable of processing packets having source and destination node addresses different from said node where the application runs, wherein:
>
> packets to be passed from the operating system to the network application are stored in a buffer and referenced via respective pointers within said first data structure, said respective pointers being exchanged between said operating system and said network application, said first data structure pointers being inserted into said first data structure by said operating system prior to network layer processing, said first data

> structure pointers being removed by said network application, insertion and removal of said first data structure pointers being asynchronous with respect to each other; and
>
> packets to be processed as received packets by said network layer of said operating system are stored in a buffer and referenced via respective pointers within said second data structure, said respective pointers being exchanged between said network application and said operating system, said second data structure pointers being inserted into said second data structure by said network application, said second data structure pointers being removed by said operating system, insertion and removal of said second data structure pointers being asynchronous with respect to each other." (emphasis added)

The test under 35 U.S.C. §103 is not whether an improvement or a use set forth in a patent would have been obvious or non-obvious; rather the test is whether the claimed invention, considered as a whole, would have been obvious. Jones v. Hardy, 110 USPQ 1021, 1024 (Fed. Cir. 1984) (emphasis added). Thus, it is impermissible to focus either on the "gist" or "core" of the invention, Bausch & Lomb, Inc. v. Barnes-Hind/Hydrocurve, Inc., 230 USPQ 416, 420 (Fed. Cir. 1986) (emphasis added). Moreover, the invention as a whole is not restricted to the specific subject matter claimed, but also embraces its properties and the problem it solves. In re Wright, 6 USPQ 2d 1959, 1961 (Fed. Cir. 1988) (emphasis added). The combination of Mahler and Garcia fails to teach the Applicants' invention as a whole.

Mahler discloses:

> To facilitate operation of the preferred embodiment, PML preferably modifies the functions of these conventional file I/O commands so as to provide a "gateway" or "PML socket" through which packets can pass between kernel space and user space. In particular, when PML application code requests the kernel to "open" a special file in a special predetermined directory, kernel code will recognize the special file as a PML file. The kernel will then allocate memory in kernel space for a file structure as usual. This file structure, however, will contain pointers to the PML kernel functions which implement the "close", "read", "write", and "ioctl" user-space function calls for this special file. In addition, the PML kernel "open" function will allocate space in the kernel for a "packet queue" to hold pointers to trapped packets. Thereafter, when PML application code seeks to "read" from that "file", a read thread created by PML kernel code will wait until a packet pointer is present in the packet queue and will

> then move the associated trapped packet into user space and delete the packet from kernel space. Similarly, when PML application code seeks to "write" a packet to the file, PML kernel code will move the packet into kernel space, and PML application code will delete the packet from user space. (see Mahler, column 7, lines 7-29).
>
> Garcia discloses:
>
> a method of constructing and using a queue for storing data elements so that it can be asynchronously managed by separate processing elements, one to sequentially add data elements to the queue, the other to sequentially remove them from the queue in the order they were added to the queue. (see Garcia, column 1, lines 55-60).

Even if the two references could somehow be operably combined, the combination would merely disclose a kernel of an operating system allocating memory in kernel space for a file structure which will hold pointers to the beginning and the end of the file on a disk, allocating memory in the kernel space for a packet queue to hold pointers to trap packets, moving the associated trapped packet into user space and deleting the packet from the kernel space, and using the queue for storing data elements so that it can be managed asynchronously by separate processing elements. In other words, the combined teachings of the cited references disclose that a packet being processed is copied from the kernel space to the user space and usually back to the kernel space.

By contrast, the Applicants' invention processes a packet in the kernel space by mapping the kernel memory where the packets reside into the user application. That is, the Applicants' invention exchanges the respective pointers of the data structure between the operating system and the network application, as opposed to copying the packet from the kernel space to the user space (see Applicants' specification, page 9, paragraph 30). Thus, the Applicants' invention provides a significant performance enhancement over the cited references, since the Applicants' invention eliminates the copying of the packet between the kernel space and the user space by exchanging the respective pointers between the operating system and the network application. Since the combined cited references fail to teach or suggest "said respective pointers being

exchanged between the operating system and said network application," the combined references fail to teach or suggest the Applicants' invention as a whole.

As such, the Applicants submit that independent claim 1 is not obvious and fully satisfy the requirements of 35 U.S.C. §103 and is patentable thereunder. Furthermore, claims 2-16 depend directly or indirectly from independent claim 1 and recite additional limitations thereof. As such and at least for the same reasons as discussed above, the Applicants submit that these dependent claims are also not obvious and fully satisfy the requirements of 35 U.S.C. §103 and are patentable thereunder. Therefore, the Applicants respectfully request that the Examiner's rejection be withdrawn.

B. Claims 17-20

Independent claim 17 (and similarly independent claim 19) recites:

> "An application programming interface (API) for network applications, which applications can process packets whose source and destination node addresses are nodes different from a node where the application runs, said API comprising:
>
> a primitive for creating a first and a second data structures associated with a specified network interface, if said data structures do not exist, and mapping said data structures both to the operating system and a specified network application, said network interface, operating system, and network application residing at a node capable of processing packets having source and destination node addresses different from said node where the application runs, wherein
>
> the specified network interface receives and sends packets from and to a network,
>
> each said packet is stored in a buffer mapped both to the operating system and the specified network application,
>
> the operating system inserts into and the specified network application removes from said first data structure, a pointer to each buffer containing a packet that the operating system's network layer outputs to the specified network interface, before the network interface sends said packets, said insertions and removals being asynchronous with respect to each other, and
>
> the specified network application inserts into and the operating system removes from said second data structure, a pointer to each buffer containing a packet that the specified network interface sends to the network, said insertions and removals being asynchronous with respect to each other." (emphasis added)

As discussed above, the combination of Mahler and Garcia discloses passing packets between the kernel space and the user space and using the queue for storing data elements so that it can be asynchronously managed by separate processing elements. Nowhere in the combined references is there any teaching or suggestion of "each said packet is stored in a buffer mapped both to the operating system and the specified network application." That is, the Applicants' invention stores the packet in a buffer that is mapped both to the operating system and the specified network application. Instead of passing copies of packets between the operating system and the network application, the kernel and user space exchange pointers to the packet (see Applicants' specification, page 9, paragraph 30). Thus, the Applicants' invention enhances performance over the cited references, since it eliminates the copying of the packets from the kernel space to the user space, and vice versa. Therefore, since the cited references fail to teach or suggest "each said packet is stored in a buffer mapped both to the operating system and the specified network application," the combined references fail to teach or suggest the Applicants' invention as a whole.

As such, the Applicants submit that independent claims 17 and 19 are not obvious and fully satisfy the requirements of 35 U.S.C. §103 and are patentable thereunder. Furthermore, claims 18 and 20 depend directly or indirectly from independent claims 17 and 19 and recite additional limitations thereof. As such and at least for the same reasons as discussed above, the Applicants submit that these dependent claims are also not obvious and fully satisfy the requirements of 35 U.S.C. §103 and are patentable thereunder. Therefore, the Applicants respectfully request that the Examiner's rejection be withdrawn.

CONCLUSION

Thus, the Applicants submit that claims 1-20 in condition for allowance. Accordingly, both reconsideration of this application and its swift passage to issue are earnestly solicited.

If, however, the Examiner believes that there are any unresolved issues requiring adverse final action in any of the claims now pending in the application, it is requested that the Examiner telephone Mr. Steven Hertzberg or Mr. Eamon Wall, telephone number (732) 530-9404, so that appropriate arrangements may be made for resolving such issues as expeditiously as possible.

Respectfully submitted,

11/22/04

E/Wall

Eamon J. Wall, Attorney
Reg. No. 39,414
(732) 530-9404

Moser, Patterson & Sheridan, LLP
595 Shrewsbury Avenue, First Floor
Shrewsbury, New Jersey 07702